ISSN: 1226-7244 (Print)
ISSN: 2288-243X (Online)
논문번호 14-04-10
68

j.inst.Korean.electr.electron.eng.Vol.18,No.4,509~516,December 2014
http://dx.doi.org/10.7471/ikeee.2014.18.4.509

고속 SDRAM에서 실시간 Matrix형 CRC
Real-time Matrix type CRC in High-Speed SDRAM

이 중 호*★

Joong-Ho Lee*★

Abstract

CRC feature in a high-speed semiconductor memory devices such as DDR4/GDDR5 increases the data reliability. Conventional CRC method have a massive area overhead and long delay time. It leads to insufficient internal timing margins for CRC calculation. This paper, presents a CRC code method that provides error detection and a real-time matrix type CRC. If there are errors in the data, proposed method can alert to the system in a real-time manner. Compare to the conventional method(XOR 6 stage ATM-8 HEC code), the proposing method can improve the error detection circuits up to 60% and XOR stage delay by 33%. Also the real-time error detection scheme can improve the error detection speed to agerage 50% for the entire data bits(UI0~UI9).

요 약

고속동작용 반도체 메모리 제품에 추가된 CRC는 DDR4와 같은 제품에서 데이타의 신뢰도를 증가시킨다. 기존의 CRC 방식은 부가회로 면적이 커고 많은 지연시간이 발생되어, CRC 계산을 위한 내부 타이밍 마진의 부족을 유발한다. 따라서 메모리 제품 설계에서 데이터 입출력 설계에 심각한 문제를 유발한다. 본 논문에서는 오류검출 회로설계를 위한 CRC 코드 방식을 제시하고, 실시간 matrix형 CRC 방법을 제안하였다. 데이터 비트오류 발생시 오류여부를 실시간으로 시스템에 피드백(feedback) 가능하도록 하였다. 제안한 방식은 기존방식(XOR 6단, ATM-8 HEC코드)대비 부가회로 면적을 60% 개선할 수 있으며, XOR 단 지연시간을 33%개선 할 수 있다. 또한 실시간 에러 검출 방식은 전체 데이터 비트(UI0~UI9)에 대해 평균 50% 이상 오류 검출 속도를 향상시켰다.

Keywords : real-time matrix type CRC, ATM-8 HEC code, DDR4, GDDR5, High-speed memory

Ⅰ. 서론

컴퓨터 시스템의 고속 동작에 따라 주기억장치로 사용되는 DRAM(Dynamic Random Access Memory)의 지속적인 동작속도의 개선이 요구되고 있다. 고속동작의 제품으로 DDR3 SDRAM 제품은 이미 상용화[1] 되었고, DDR4 SDRAM이 상용화 단계에 있다.[2] 뿐만 아니라 그래픽제품 전용 GDDR5도 고속동작 목적으로 개발된 제품이다. 컴퓨터 시스템이 고속화 및 저전력화 될수록 데이터의 유효폭(tDV : Data Vaild window)의 확보가 어려워지며, 시스템간에 전송된 데이터에서 오류가 발생할 확률이 증가한다.[3] 이러한 문제를 해결하기위해 DDR4나 GDDR5 제품에서는 CRC(Cyclic Redundancy Check)와 DBI(Data Bus Inversion) 기능[3-4]을 추가하여 데이터 전송 오류시 오류 검출 여부를 시스템에 전송(Alert 신호발생)하여 데이터를 다시 전송받을 수 있도록 하였다. CRC구현을 위해 DDR4와 GDDR5에서는 동일한 8비트-CRC

*★ Dept. of Computer Science, Yongin University
joongho65@yongin.ac.kr TEL: 031-8020-2768
Manuscript received Oct. 20, 2014; revised Dec. 1, 2014 ; accepted Dec. 3. 2014

코드를 사용하며, ATM-8 HEC 코드로서 다항식 x^8+x^2+x+1을 사용한다.[1,5-8]

CRC 코드 구현시 시스템에서 데이터 오류검사를 위해 부가적인 시간과 칩 면적이 요구된다. CRC로 인한 부가적인 시간은 DRAM에 접근시간(access time)을 감소시키는 주요 요소중 하나이다. DDR4 CRC 동작시간에서 데이터가 연속적으로 전송되기 위해 tCCD(CAS to CAS delay time)을 초과하지 않아야하며, 요구되는 제한 시간은 아래와 같다.[5]

$$\text{CRC time} < \text{tCCD} = 5\text{nCK} \quad (1)$$

tCCD 항목에서 데이터전송 명령(Read)이후 메모리 Core로부터 GIO(Global IO line)에 데이터를 실어서 DQ pin으로 보내는 경로 중에 CRC 결과를 만들어서 데이터에 첨부하는 일련의 과정이 연속적(serial)으로 일어난다. 메모리 Core로부터 연속적으로 데이터가 전송되어 DQ pin으로 최종 전송되기 위해 전통 SDRAM에서 요구되는 제한 시간이외에 CRC결과를 만들기 위한 시간이 추가되며 이러한 과정은 최소 CL(CAS Latency)를 초과하게 하는 요인이 된다. CRC를 만들기 위해 내부 마진(margin)까지 고려하면, 3.2Gbps 제품의 경우, burst 8데이타(UI0~UI7)를 전송할 동안 CRC를 생성하여 8UI에 실어야한다. 이때 1.406ns 이내에 CRC가 만들어져야 하며 CRC 계산에 허용되는 XOR 지연 시간은 120ps이다.[5] ATM-8 HEC코드는 6단 XOR구성되어, 3.2Gbps 제품에서 120ps를 만족하기 위한 20n급 공정 PVT조건은 너무 열악하며, 고속화 및 저전력화 될수록 더욱 악화된다. 따라서 XOR 게이트의 지연시간을 줄이기 위한 구성방법에 관한 연구도 진행되었다.[9,10]

한편, ATM-8 HEC 코드 다항식 x^8+x^2+x+1을 구성하기 위한 데이터 비트와 패리피 비트는 각 데이터 입출력 핀인 DQ0~DQ7 및 DBI의 UI0~UI9까지의 데이터들로 구성되며, UI(Unit Interval)는 BL(burst length)를 구성하는 최소 단위이다. JEDEC(Joint Electron for Devices Engineering Council)에 표준화된 BL8에 대한 데이터 프레임(data frame)은 그림 1에 나타내었으며, DDR4 spec.에 기초한 코드워드 생성을 위한 데이트 비트의 CRC구성을 아래에 나타내었다.[11] 그림 1에서 72비트의 데이터에 대해 8비트 CRC를 구현하기 위해 272개의 2-input XOR 게이트가 요구되며, 6단 XOR로 구성된다. 그림에서 UI0~UI8까지 모든 데이터가 입력된 후에 CRC 검사가 가능하며, DDR4에서 tCCD(CAS to CAS delay=5nCK, UI0~UI9)이내에 오류 검사가 완료되어야 한다. 5nCK는 3.2Gbps 제품의 경우 1.56ns시간이며 8UI에 CRC비트가 있으므로 1.406ns이후에 오류검사가 가능하다. 결론적으로 데이터 비트에 오류가 포함된 경우 1.406ns가 경과된 이후에 시스템에 오류 여부를 알려주는 Alert 신호를 보낼 수 있다.

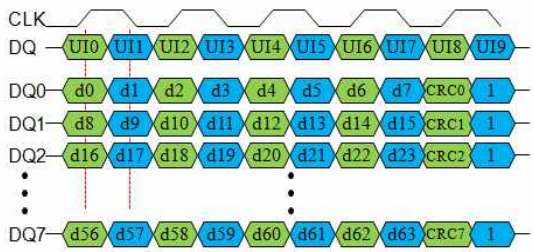


Fig. 1. the configuration of 8UI DQ data and the CRC for DDR4

그림 1. DDR4에 대한 8UI DQ 데이터와 CRC의 구성

$$\begin{aligned}\text{CRC0} = {} & \text{d69} \oplus \text{d68} \oplus \text{d67} \oplus \text{d66} \oplus \text{d64} \oplus \text{d63} \oplus \text{d60} \oplus \text{d56} \oplus \text{d54} \oplus \\ & \text{d53} \oplus \text{d52} \oplus \text{d50} \oplus \text{d49} \oplus \text{d48} \oplus \text{d45} \oplus \text{d43} \oplus \text{d40} \oplus \text{d39} \oplus \\ & \text{d35} \oplus \text{d34} \oplus \text{d31} \oplus \text{d30} \oplus \text{d28} \oplus \text{d23} \oplus \text{d21} \oplus \text{d19} \oplus \text{d18} \oplus \\ & \text{d16} \oplus \text{d14} \oplus \text{d12} \oplus \text{d8} \oplus \text{d7} \oplus \text{d6} \oplus \text{d0}\end{aligned}$$

$$\begin{aligned}\text{CRC1} = {} & \text{d70} \oplus \text{d66} \oplus \text{d65} \oplus \text{d63} \oplus \text{d61} \oplus \text{d60} \oplus \text{d57} \oplus \text{d56} \oplus \text{d55} \oplus \\ & \text{d52} \oplus \text{d51} \oplus \text{d48} \oplus \text{d46} \oplus \text{d45} \oplus \text{d44} \oplus \text{d43} \oplus \text{d41} \oplus \text{d39} \oplus \\ & \text{d36} \oplus \text{d34} \oplus \text{d32} \oplus \text{d30} \oplus \text{d29} \oplus \text{d28} \oplus \text{d24} \oplus \text{d23} \oplus \text{d22} \oplus \\ & \text{d21} \oplus \text{d20} \oplus \text{d18} \oplus \text{d17} \oplus \text{d16} \oplus \text{d15} \oplus \text{d14} \oplus \text{d13} \oplus \text{d12} \oplus \\ & \text{d9} \oplus \text{d6} \oplus \text{d1} \oplus \text{d0}\end{aligned}$$

$$\begin{aligned}\text{CRC2} = {} & \text{d71} \oplus \text{d69} \oplus \text{d68} \oplus \text{d63} \oplus \text{d62} \oplus \text{d61} \oplus \text{d60} \oplus \text{d58} \oplus \text{d57} \oplus \\ & \text{d54} \oplus \text{d50} \oplus \text{d48} \oplus \text{d47} \oplus \text{d46} \oplus \text{d44} \oplus \text{d43} \oplus \text{d42} \oplus \text{d39} \oplus \\ & \text{d37} \oplus \text{d34} \oplus \text{d33} \oplus \text{d29} \oplus \text{d28} \oplus \text{d25} \oplus \text{d24} \oplus \text{d22} \oplus \text{d17} \oplus \\ & \text{d15} \oplus \text{d13} \oplus \text{d12} \oplus \text{d10} \oplus \text{d8} \oplus \text{d6} \oplus \text{d2} \oplus \text{d1} \oplus \text{d0}\end{aligned}$$

• • •

$$\begin{aligned}\text{CRC7} = {} & \text{d68} \oplus \text{d67} \oplus \text{d66} \oplus \text{d65} \oplus \text{d63} \oplus \text{d62} \oplus \text{d59} \oplus \text{d55} \oplus \text{d53} \oplus \\ & \text{d52} \oplus \text{d51} \oplus \text{d49} \oplus \text{d48} \oplus \text{d47} \oplus \text{d44} \oplus \text{d42} \oplus \text{d39} \oplus \text{d38} \oplus \\ & \text{d34} \oplus \text{d33} \oplus \text{d30} \oplus \text{d29} \oplus \text{d27} \oplus \text{d22} \oplus \text{d20} \oplus \text{d18} \oplus \text{d17} \oplus \\ & \text{d15} \oplus \text{d11} \oplus \text{d7} \oplus \text{d6} \oplus \text{d5}\end{aligned}$$

본 연구에서는 CRC 를 만드는데 요구되는 시간 제약을 개선하기 위해 matrix 방식의 오류검출이 가능한 코드구성 방식을 제안하였고, CRC 오류검사 결과

를 빠르게 시스템에 피드백(feedback) 주기위해 실시간 피드백이 가능한 CRC 검사 방법을 제시하였다. 즉 데이터 오류가 나타나는 즉시 Alert 신호를 발생시켜 시스템에 데이터 전송을 멈추고 재전송을 요구함으로써 시스템 성능을 개선할 수 있도록 하였다. 또한 데이터 비트 오류 위치와 오류 비트의 수에 따른 성능개선 정도를 분석하였다. 제안하는 방식은 DDR4의 경우를 예로 보인 것이며 CRC를 채용하는 고속 SDRAM에 적용가능하다.

II. CRC 구성

실시간 오류 데이터 검출이 가능하도록 하기위해 그림 1의 데이터 프레임에서 CRC를 재구성하여 그림 2와 같이 변경 구성토록 제안한다. 기존의 데이터 프레임 대비 CRC에 해당하는 UI8, 9 비트를 UI0, 1로 이동시켜 데이터 보다 CRC 비트를 먼저 입력 받는다. CRC비트의 위치를 맨 앞으로 가져와도 데이터의 처리순서는 변경되지 않으며, 각 DQ의 전체 burst에 대해 UI0~UI9까지 처리가 완료되어야 하므로 데이터의 처리 시간에 영향을 주지 않는 조건을 만족한다. 데이터 프레임 구성방식의 제안은 차세대 고속 메모리 제품에서 적용가능하다.

UI	0	1	2	3	4	5	6	7	8	9
DQ0	CRC0	CRC8	d0	d1	d2	d3	d4	d5	d6	d7
DQ1	CRC1	CRC9	d8	d9	d10	d11	d12	d13	d14	d15
DQ2	CRC2	CRC10	d16	d17	d18	d19	d20	d21	d22	d23
DQ3	CRC3	CRC11	d24	d25	d26	d27	d28	d29	d30	d31
DQ4	CRC4	CRC12	d32	d33	d34	d35	d36	d37	d38	d39
DQ5	CRC5	CRC13	d40	d41	d42	d43	d44	d45	d46	d47
DQ6	CRC6	CRC14	d48	d49	d50	d51	d52	d53	d54	d55
DQ7	CRC7	CRC15	d56	d57	d58	d59	d60	d61	d62	d63

Fig. 2. proposed a real-time 8UI DQ data and the CRC configuration

그림 2. 제안한 실시간 8UI DQ 데이터와 CRC의 구성

1. 오류 검출을 위한 CRC 다항식(polynomial)구성

기존의 ATM-8 HEC코드 구성을 변경하여 그림 2의 데이터 프레임으로부터 새로운 CRC 검사비트를 구성하여 아래에 나타내었다. CRC 비트는 크게 두분류로 CRC0~CRC7과 CRC8~CRC15로 구성되는데, CRC0~CRC7는 실시간 오류를 검출목적으로 사용된다. CRC8~CRC15는 실시간 오류검출은 불가하나 앞의 실시간 오류 검출 비트의 오류 검출범위를 보완하기 위해 추가되었다. 모든 DQ에서 UI0에 대한 모든 데이터의 오증(syndrome) S0를 구성하여 CRC0, UI1의 모든 데이터의 오증 S1을 구성하여 CRC1과 같이 구성하며, CRC7까지 8비트 CRC를 완성한다. 정상 데이타의 경우 S0~S7은 모두 논리 "0"이며, 오류 데이터가 포함된 경우 해당 오증다항식은 논리 "1"값을 가지며 오류가 발생했음을 나타낸다. 따라서 임의의 DQ에 대해 첫 번째 UI(UI2) 데이터에 오류발생시 S0가 논리 "1"이 되어 즉시 오류 여부를 피드백 가능하다. 마찬가지로 두 번째 UI(UI3)에 오류 발생시 S1이 논리 "1"값을 가지며, UI3~UI9까지의 데이터 오류에 대해서 동일한 방식으로 오류 검출이 되며 burst 데이터가 입력되는 매 클럭 사이클마다 오류 여부를 피드백가능하다. 또한 기존의 UI9는 사용하지 않아서 논리값 "1"로 고정하였지만, 본 논문에서는 이를 활용하도록 CRC 비트로 사용하여(CRC8~CRC15) S8~S15의 오증다항식을 가지도록 하여 오류 검출율을 개선하도록 하였다. CRC8은 DQ0의 burst 데이터를 검사할 수 있도록 추가한 패리티 비트이며 CRC9는 DQ1에 대한 패리티 비트이고 DQ2~DQ7까지 동일한 방식으로 패리티 비트를 추가하였다. 즉 CRC8~CRC15는 CRC0~CRC7에서 검출하지 못하는 이중고장(double error)를 검출하기 위해 추가되었다. 4절 표 4에서 제안한 방식의 오류 검출범위를 나타내었는데 여기서 상세히 설명하였다.

$$d0 \oplus d8 \oplus d16 \oplus d24 \oplus d32 \oplus d40 \oplus d48 \oplus d56 \oplus CRC0 = S0$$
$$d1 \oplus d9 \oplus d17 \oplus d25 \oplus d33 \oplus d41 \oplus d49 \oplus d57 \oplus CRC1 = S1$$
$$d2 \oplus d10 \oplus d18 \oplus d26 \oplus d34 \oplus d42 \oplus d50 \oplus d58 \oplus CRC2 = S2$$
$$d3 \oplus d11 \oplus d19 \oplus d27 \oplus d35 \oplus d43 \oplus d51 \oplus d59 \oplus CRC3 = S3$$
$$d4 \oplus d12 \oplus d20 \oplus d28 \oplus d36 \oplus d44 \oplus d52 \oplus d60 \oplus CRC4 = S4$$
$$d5 \oplus d13 \oplus d21 \oplus d29 \oplus d37 \oplus d45 \oplus d53 \oplus d61 \oplus CRC5 = S5$$
$$d6 \oplus d14 \oplus d22 \oplus d30 \oplus d38 \oplus d46 \oplus d54 \oplus d62 \oplus CRC6 = S6$$
$$d7 \oplus d15 \oplus d23 \oplus d31 \oplus d39 \oplus d47 \oplus d55 \oplus d63 \oplus CRC7 = S7$$

$$d0 \oplus d1 \oplus d2 \oplus d3 \oplus d4 \oplus d5 \oplus d6 \oplus d7 \oplus CRC8 = S8$$
$$d8 \oplus d9 \oplus d10 \oplus d11 \oplus d12 \oplus d13 \oplus d14 \oplus d15 \oplus CRC9 = S9$$
$$d16 \oplus d17 \oplus d18 \oplus d19 \oplus d20 \oplus d21 \oplus d22 \oplus d23 \oplus CRC10 = S10$$
$$d24 \oplus d25 \oplus d26 \oplus d27 \oplus d28 \oplus d29 \oplus d30 \oplus d31 \oplus CRC11 = S11$$
$$d32 \oplus d33 \oplus d34 \oplus d35 \oplus d36 \oplus d37 \oplus d38 \oplus d39 \oplus CRC12 = S12$$
$$d40 \oplus d41 \oplus d42 \oplus d43 \oplus d44 \oplus d45 \oplus d46 \oplus d47 \oplus CRC13 = S13$$
$$d48 \oplus d49 \oplus d50 \oplus d51 \oplus d52 \oplus d53 \oplus d54 \oplus d55 \oplus CRC14 = S14$$
$$d56 \oplus d57 \oplus d58 \oplus d59 \oplus d60 \oplus d61 \oplus d62 \oplus d63 \oplus CRC15 = S15$$

2. 실시간 오류검출 회로구성 및 원리

구성한 오증다항식에 실시간 오류검출 기능을 추가하기 위해 로직을 새로이 구성하여 그림 3에 S0의 예를 나타내었으며, S1~S7까지 동일한 방식으로 구성한다. 그림 3에서 전체 데이터를 이분하여, E1은 데이터 d0, d8, d16, d24,를 E2는 d32, d40, d48, d56을 관장하여 오류검출 여부를 검사하기 위한 회로중간 노드이다. S8~S15는 burst 데이터 모두가 입력된 후 UI9 이후에 데이터 오류 여부를 판정 가능하여, 실시간 오류 검출기능의 의미가 없다. 따라서 실시간 오류검출기능을 적용하지 않으며 단지 오류 검출기능만 가진다.

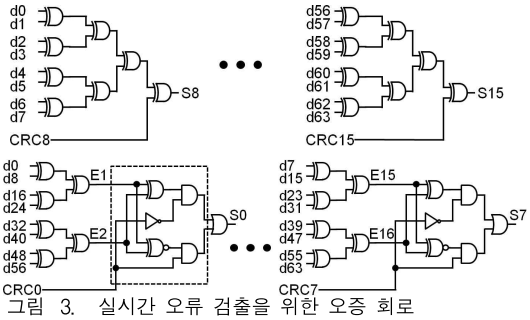


그림 3. 실시간 오류 검출을 위한 오증 회로
Fig. 3. syndrome circuits for real-time error detection

그림 3에서 처럼 S0 구성은 E1과 E2로 이분하여 나타낼 수 있고, 오증다항식을 구성하는 모든 데이터의 논리합이 "0"이 되어야하기 때문에 E1과 E2의 논리합이 "0" 이어야 함을 의미한다. 표 1에 E1과 E2로 이분할 경우 나타날 수 있는 데이터 패턴에 따라 논리 "1"개수의 조합을 나타내었다. 데이터가 어떤 패턴을 가지든지 결론적으로 총 4가지의 경우("정상 데이타 1~4")로 나타난다. 따라서 E1, E2와 CRC 패리티 비트 조합에 의해 데이터 오류 여부를 판별할 수 있다. 표 1은 S0에 대한 예를 나타낸 것으로, 데이터에 오류가 없을 경우, d0, d8, d16, d24 데이터의 "1"의 개수가 짝수이면 E1은 "0"이고, 홀수이면 "1"이다. 만약 E1이 논리 "1", E2가 논리 "1"이고 CRC0이 논리 "0"이면 데이터 오류가 없음을 나타내며 S0는 논리 "0"이 된다. 이것은 E1과 E2를 보면 CRC값을 결정할 수 있다는 것을 나타낸다. 만약 데이터에 오류가 있는 경우 "정상 데이터 1~4"의 각 경우에 따라 오류가 발생할 수 있는 경우의 조합을 표에서 처럼 "1-1"~"4-2"까지 분류하여 나타낼 수 있다. "정상 데이터 1"의 경우로 부터 E1에서 오류가 있을 경우는 "오류데이터 1-1", E2에 오류가 있을 경우 "오류 데이터 1-2"의 경우로 나타나는데, 데이터 1의 개수가 홀수가 아닌 짝수로 나타나며 최종적으로 각각의 경우 E1과 E2의 논리합은 "1"이 된다. 이때 CRC0는 "0"값을 가지므로 XOR를 거쳐서 오증다항식 결과는 "1"이 되어 오류가 발생했음을 나타낸다. 동일한 방식으로 "정상 데이타 2"로 부터 E2에서 오류가 있을 경우("오류 데이터 2-2") 데이터의 "1"의 개수가 홀수가 되어 E1과 E2의 논리합이 "1"이 되며 이때 CRC는 "1"이므로 그림 3에서 XOR는 "0"을 출력하지만, XNOR는 "1"이 되어 S0가 "1"이 되므로 오류가 발생했음을 확인 할 수 있다. 이와 같이 모든 오증다항식에 대해 동일한 원리로 회로를 구성할 수 있다.

Table 1. combination of normal data and error data set
표 1. 정상 데이타와 오류 데이터 세트의 조합

데이터 분류			E1		E2		CRC	S0
			d0,d8,d16,d24		d32,d40,d48,d56		CRC0	SUM
정상 데이터 조합		1	odd	1	odd	1	0	0
		2	odd	1	even	0	1	0
		3	even	0	odd	1	1	0
		4	even	0	even	0	0	0
오류 데이터 조합	1	1-1	even	0	odd	1	0	1
		1-2	odd	1	even	0		
	2	2-1	even	0	even	0	1	1
		2-2	odd	1	odd	1		
	3	3-1	odd	1	odd	1	1	1
		3-2	even	0	even	0		
	4	4-1	odd	1	even	0	0	1
		4-2	even	0	odd	1		

3. 실시간 오류검출 알고리즘

제안한 방식의 시간이 흐름에 따른 실시간 오류검출 과정을 그림 4에 나타내었다. 만약 첫 번째 데이터(UI2)에서 오류가 발생했을 경우, UI0에서 입력된 CRC값과 DQ0~DQ7의 각 데이타를 sum하여 결과가 "1"이면 오류가 발생했으므로 Alert 신호를 발생하여 시스템으로 전송한다. 본 방식은 매 클럭 사이클에 발생되는 데이터의 오류를 실시간으로 시스템에 피드백 가능하다. 그림 4에서 처럼 데이터 오류가 첫 번째 클럭 사이클(UI2)에서 발생할 경우 피드백 시간의 개선이 87.5%로 가장 뛰어나며, UI2이후로 UI가 증가할수록 시간 개선량이 감소한다.

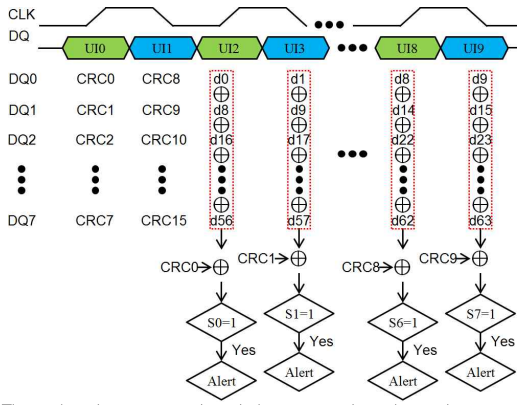
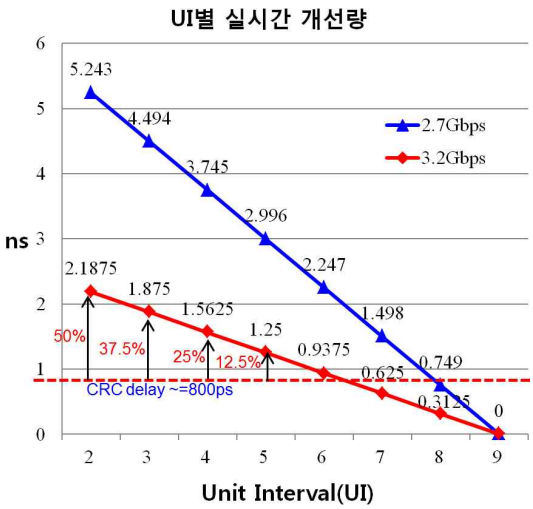


Fig. 4. the proposed real-time error detection scheme
그림 4. 제안한 실시간 오류 검출 scheme

실시간 피드백에 따른 DDR 3200과 2667의 경우 UI별 개선정도를 그림 5에 나타내었다. 그림 5는 오류검사회로의 지연시간을 고려하지 않았으며, 이를 고려하면 개선량이 감소하는데, UI9에서는 오히려 시간 손해가 발생한다. 따라서 오류검사 회로 지연시간과 UI별 개선량에 따른 전체 개선량을 고려한 trade-off 설계가 필요하다. 3.2Gbps (tCK=0.625ns) 동작제품에서 ~800ps 동작 XOR 회로를[1,5] 적용할 경우 UI2에서 50% 시간 개선효과가 있고, UI5에서 12.5%의 개선효과가 있으며, UI5 이후로는 시간 손해



UI	UI별 feedback 시간 개선량									
	0	1	2	3	4	5	6	7	8	9
DQ	-	-	87.5%	75%	62.5%	50%	37.5%	25%	12.5%	0%

Fig. 5. In real-time error feedback on the amount of UI improvements
그림 5. 실시간 오류 피드백시 UI별 개선량

가 발생한다. 반면 DDR 2667(2.7 Gbps) 제품에서는 UI7까지 시간 개선효과가 있다.

4. 실시간 오류검출 범위 및 전체 오류 검출 결과

실시간 오류 검출범위 산출을 위해 오류비트 발생 가능 경우에 대한 분류가 선행되어야 하기에 오류 유형별로 분류한다. 10UI에 대한 8개의 DQ(DQ0~DQ7)에서 단일비트의 오류(single bit error)가 발생할 수 있는 조합의 수는 C(80,1)=80개 이고, 이중오류(double bit error)가 발생할 수 있는 조합은 C(80,2)=3,160개 이다. 삼중오류(triple bit error)는 C(80,3)=82,160개의 조합을 가진다. 전체 데이터 비트 10개중 9비트는 정상(pass), 1비트는 오류(fail)를 p9f1로 표기하며, 오류 비트의 위치를 p9f1(x)로 표기한다. 이러한 방법으로 단일비트 오류에 대한 경우의 수를 표 2에 수식으로 간략화하여 나타내었다. DQ0에서 단일 비트오류가 나타날 수 있는 경우의 수는 총 10개의 UI에 대해 10개의 경우의 수가 존재한다.

Table 2. a number of single-bit errors per each DQ
표 2. 각 DQ별 단일비트 오류의 경우의 수

Number of Combination for Single Bit Errors		
DQ0	all UI	$\cup_{UI=0}^{9} p^9 f^1(UI)$ = 10
DQ1	all UI	$\cup_{UI=0}^{9} p^9 f^1(UI)$ = 10
...	...	...
DQ7	all UI	$\cup_{UI=0}^{9} p^9 f^1(UI)$ = 10
Total(all DQ, all UI)		$\cup_{DQ=0}^{7}\{\cup_{UI=0}^{9} p^9 f^1(UI)\}(DQ)$ = 8X10 = 80

표 2의 단일 비트오류 이외의 이중 비트오류에 대해 표3에 정리하여 나타내었다. 표 3의 Type 1에 각각의 UI에서 모든 DQ에 대해 이중오류가 발생 가능한 경우를 나타내었는데, 1항의 경우 DQ0에서 오류가 발생하고, 나머지 DQ에서 한 비트 오류가 발생하는 이중오류를 나타낸 예이다. 즉 p6f2(0,2)는 DQ0과 DQ2에서 각 한 비트씩 오류가 발생한 경우이며, p6f2{(0,1),(0,2),...,(0,7)}은 DQ0에 한 비트, 나머지 DQ(1~7)에서 비트 오류가 발생 가능한 조합의 합집합을 나타낸 것이다. 이것을 모든 UI에 대해 확장하면 총 70개의 경우의 수가 발생한다. 따라서 고정된 UI에 대해 이중 고장의 모든 경우의 수는 Type 1의 1~7까지의 경우의 수이며, 총 280가지가 발생된다. 이는 8개의 DQ에 대해 2비트 오류가 발생가능한 조합의 수 C(8,2)=28에서 UI10에 대한 경우의 합과 일치한다.

Type 2의 1항은 UI0에 오류가 있고 나머지 UI에

Table 3. number of double-bit errors per each DQ
표 3. 각 DQ별 이중비트 오류의 경우의 수

Number of Combination for Double Bit Errors		
#	Type 1	
1	$p^6f^2((0,1),(0,2),\dots,(0,7))$	$\cup_{DQ=1}^{7}p^6f^2(0,DQ)$ = 7, $\cup_{UI=0}^{9}\{\cup_{DQ=1}^{7}p^6f^2(0,DQ)\}(UI)$ = 10X7
2	$p^6f^2((1,2),(1,3),\dots,(1,7))$	$\cup_{DQ=2}^{7}p^6f^2(1,DQ)$ = 6, $\cup_{UI=0}^{9}\{\cup_{DQ=2}^{7}p^6f^2(1,DQ)\}(UI)$ = 10X6
3	$p^6f^2((2,3),(2,4),\dots,(2,7))$	$\cup_{DQ=3}^{7}p^6f^2(2,DQ)$ = 5, $\cup_{UI=0}^{9}\{\cup_{DQ=3}^{7}p^6f^2(2,DQ)\}(UI)$ = 10X5
4	$p^6f^2((3,4),(3,5),\dots,(3,7))$	$\cup_{DQ=4}^{7}p^6f^2(3,DQ)$ = 4, $\cup_{UI=0}^{9}\{\cup_{DQ=4}^{7}p^6f^2(3,DQ)\}(UI)$ = 10X4
5	$p^6f^2((4,5),(4,6),(4,7))$	$\cup_{DQ=5}^{7}p^6f^2(4,DQ)$ = 3, $\cup_{UI=0}^{9}\{\cup_{DQ=5}^{7}p^6f^2(4,DQ)\}(UI)$ = 10X3
6	$p^6f^2((5,6),(5,7))$	$\cup_{DQ=6}^{7}p^6f^2(5,DQ)$ = 2, $\cup_{UI=0}^{9}\{\cup_{DQ=6}^{7}p^6f^2(5,DQ)\}(UI)$ = 10X2
7	$p^6f^2((6,7))$	$\cup_{DQ=7}^{7}p^6f^2(6,DQ)$ = 5, $\cup_{UI=0}^{9}\{\cup_{DQ=7}^{7}p^6f^2(6,DQ)\}(UI)$ = 10X1
Total(all DQ, all UI)		$\cup_{DQ=0}^{9}\{C(8,2)\}$ = 10X28 = 280

Number of Combination for Double Bit Errors		
#	Type 2	Type 3
1	$\cup_{DQ=0}^{7}\{\cup_{UI=1}^{9}p^8f^2(0,UI)\}(DQ)$	$\cup_{UI=1}^{9}\{\cup_{DQ=1}^{7}p^6f^2(0,UI,DQ)\}$
2	$\cup_{DQ=0}^{7}\{\cup_{UI=2}^{9}p^8f^2(1,UI)\}(DQ)$	$\cup_{UI=1}^{9}\{\cup_{DQ=0,\neq1}^{7}p^6f^2(1,UI,DQ)\}$
3	$\cup_{DQ=0}^{7}\{\cup_{UI=3}^{9}p^8f^2(2,UI)\}(DQ)$	$\cup_{UI=1}^{9}\{\cup_{DQ=0,\neq2}^{7}p^6f^2(2,UI,DQ)\}$
4	$\cup_{DQ=0}^{7}\{\cup_{UI=4}^{9}p^8f^2(3,UI)\}(DQ)$	$\cup_{UI=1}^{9}\{\cup_{DQ=0,\neq3}^{7}p^6f^2(3,UI,DQ)\}$
5	$\cup_{DQ=0}^{7}\{\cup_{UI=5}^{9}p^8f^2(4,UI)\}(DQ)$	$\cup_{UI=1}^{9}\{\cup_{DQ=0,\neq4}^{7}p^6f^2(4,UI,DQ)\}$
6	$\cup_{DQ=0}^{7}\{\cup_{UI=6}^{9}p^8f^2(5,UI)\}(DQ)$	$\cup_{UI=1}^{9}\{\cup_{DQ=0,\neq5}^{7}p^6f^2(5,UI,DQ)\}$
7	$\cup_{DQ=0}^{7}\{\cup_{UI=7}^{9}p^8f^2(6,UI)\}(DQ)$	$\cup_{UI=1}^{9}\{\cup_{DQ=0,\neq6}^{7}p^6f^2(6,UI,DQ)\}$
8	$\cup_{DQ=0}^{7}\{\cup_{UI=8}^{9}p^8f^2(7,UI)\}(DQ)$	$\cup_{UI=1}^{9}\{\cup_{DQ=0,\neq7}^{7}p^6f^2(7,UI,DQ)\}$
9	$\cup_{DQ=0}^{7}\{\cup_{UI=9}^{9}p^8f^2(8,UI)\}(DQ)$	-
Total	$\cup_{DQ=0}^{7}\{C(10,2)\}$ = 8X45 = 360	8X$\{\cup_{UI=1}^{9}\{C(7,1)\}\}$= 8X{9X7}=504

#	Type 4	Type 5
1	$\cup_{UI=2}^{9}\{\cup_{DQ=1}^{7}p^6f^2(0,UI,DQ)\}$	$\cup_{UI=3}^{9}\{\cup_{DQ=1}^{7}p^6f^2(0,UI,DQ)\}$
2	$\cup_{UI=2}^{9}\{\cup_{DQ=0,\neq1}^{7}p^6f^2(1,UI,DQ)\}$	$\cup_{UI=3}^{9}\{\cup_{DQ=0,\neq1}^{7}p^6f^2(1,UI,DQ)\}$
3	$\cup_{UI=2}^{9}\{\cup_{DQ=0,\neq2}^{7}p^6f^2(2,UI,DQ)\}$	$\cup_{UI=3}^{9}\{\cup_{DQ=0,\neq2}^{7}p^6f^2(2,UI,DQ)\}$
4	$\cup_{UI=2}^{9}\{\cup_{DQ=0,\neq3}^{7}p^6f^2(3,UI,DQ)\}$	$\cup_{UI=3}^{9}\{\cup_{DQ=0,\neq3}^{7}p^6f^2(3,UI,DQ)\}$
5	$\cup_{UI=2}^{9}\{\cup_{DQ=0,\neq4}^{7}p^6f^2(4,UI,DQ)\}$	$\cup_{UI=3}^{9}\{\cup_{DQ=0,\neq4}^{7}p^6f^2(4,UI,DQ)\}$
6	$\cup_{UI=2}^{9}\{\cup_{DQ=0,\neq5}^{7}p^6f^2(5,UI,DQ)\}$	$\cup_{UI=3}^{9}\{\cup_{DQ=0,\neq5}^{7}p^6f^2(5,UI,DQ)\}$
7	$\cup_{UI=2}^{9}\{\cup_{DQ=0,\neq6}^{7}p^6f^2(6,UI,DQ)\}$	$\cup_{UI=3}^{9}\{\cup_{DQ=0,\neq6}^{7}p^6f^2(6,UI,DQ)\}$
8	$\cup_{UI=2}^{9}\{\cup_{DQ=0,\neq7}^{7}p^6f^2(7,UI,DQ)\}$	$\cup_{UI=3}^{9}\{\cup_{DQ=0,\neq7}^{7}p^6f^2(7,UI,DQ)\}$
9	-	-
Total	8X$\{\cup_{UI=2}^{9}\{C(7,1)\}\}$= 8X{8X7}=448	8X$\{\cup_{UI=3}^{9}\{C(7,1)\}\}$= 8X{7X7}=392

#	...	Type 10	Type 11
1		$\cup_{UI=8}^{9}\{\cup_{DQ=1}^{7}p^6f^2(0,UI,DQ)\}$	$\cup_{UI=9}^{9}\{\cup_{DQ=1}^{7}p^6f^2(0,UI,DQ)\}$
2		$\cup_{UI=8}^{9}\{\cup_{DQ=0,\neq1}^{7}p^6f^2(1,UI,DQ)\}$	$\cup_{UI=9}^{9}\{\cup_{DQ=0,\neq1}^{7}p^6f^2(1,UI,DQ)\}$
3		$\cup_{UI=8}^{9}\{\cup_{DQ=0,\neq2}^{7}p^6f^2(2,UI,DQ)\}$	$\cup_{UI=9}^{9}\{\cup_{DQ=0,\neq2}^{7}p^6f^2(2,UI,DQ)\}$
4		$\cup_{UI=8}^{9}\{\cup_{DQ=0,\neq3}^{7}p^6f^2(3,UI,DQ)\}$	$\cup_{UI=9}^{9}\{\cup_{DQ=0,\neq3}^{7}p^6f^2(3,UI,DQ)\}$
5	...	$\cup_{UI=8}^{9}\{\cup_{DQ=0,\neq4}^{7}p^6f^2(4,UI,DQ)\}$	$\cup_{UI=9}^{9}\{\cup_{DQ=0,\neq4}^{7}p^6f^2(4,UI,DQ)\}$
6		$\cup_{UI=8}^{9}\{\cup_{DQ=0,\neq5}^{7}p^6f^2(5,UI,DQ)\}$	$\cup_{UI=9}^{9}\{\cup_{DQ=0,\neq5}^{7}p^6f^2(5,UI,DQ)\}$
7		$\cup_{UI=8}^{9}\{\cup_{DQ=0,\neq6}^{7}p^6f^2(6,UI,DQ)\}$	$\cup_{UI=9}^{9}\{\cup_{DQ=0,\neq6}^{7}p^6f^2(6,UI,DQ)\}$
8		$\cup_{UI=8}^{9}\{\cup_{DQ=0,\neq7}^{7}p^6f^2(7,UI,DQ)\}$	$\cup_{UI=9}^{9}\{\cup_{DQ=0,\neq7}^{7}p^6f^2(7,UI,DQ)\}$
9		-	-
Total	...	8X$\{\cup_{UI=8}^{9}\{C(7,1)\}\}$= 8X{2X7}=112	8X$\{\cup_{UI=9}^{9}\{C(7,1)\}\}$= 8X{1X7}=56

Table 4. double-bit error detection rate for real-time scheme
표 4. 제안한 실시간 이중비트 오류 검출율

오류검출 CRC 및 실시간 오류검출 여부						
#	Type 5		Type 6		Type 7	
	오류검출	실시간	오류검출	실시간	오류검출	실시간
1	(CRC2,3),(8,9),...,(8,15)	O	(CRC3,4),(8,9),...,(8,15)	O	(CRC4,5),(8,9),...,(8,15)	O
2	(CRC2,4),(8,9),...,(8,15)	O	(CRC3,5),(8,9),...,(8,15)	O	(CRC4,6),(8,9),...,(8,15)	O
3	(CRC2,5),(8,9),...,(8,15)	O	(CRC3,6),(8,9),...,(8,15)	O	(CRC4,7),(8,9),...,(8,15)	O
4	(CRC2,6),(8,9),...,(8,15)	O	(CRC3,7),(8,9),...,(8,15)	O	(CRC4,8),(8,9),...,(8,15)	O
5	(CRC2,7),(8,9),...,(8,15)	O	(CRC3,8),(8,9),...,(8,15)	O	(CRC4,9),(8,9),...,(8,15)	O
6	(CRC2,8),(8,9),...,(8,15)	O	(CRC3,9),(8,9),...,(8,15)	O	-	-
7	(CRC2,9),(8,9),...,(8,15)	O	-	-	-	-
8	-	-	-	-	-	-
%	12.40%		10.63%		8.86%	

오류검출 CRC 및 실시간 오류검출 여부								
#	Type 8		Type 9		Type 10		Type 11	
	오류검출	실시간	오류검출	실시간	오류검출	실시간	오류검출	실시간
1	(CRC5,6),(CRC8,9),...,(8,15)	O	(CRC6,7),(CRC8,9),...,(8,15)	O	(CRC7,8),(CRC8,9),...,(8,15)	O	(CRC8,9),(CRC8,9),...,(8,15)	O
2	(CRC5,7),(CRC8,9),...,(8,15)	O	(CRC6,8),(CRC8,9),...,(8,15)	O	(CRC7,9),(CRC8,9),...,(8,15)	O	-	-
3	(CRC5,8),(CRC8,9),...,(8,15)	O	(CRC6,9),(CRC8,9),...,(8,15)	O	-	-	-	-
4	(CRC5,9),(CRC8,9),...,(8,15)	O	-	-	-	-	-	-
5	-	-	-	-	-	-	-	-
6	-	-	-	-	-	-	-	-
7	-	-	-	-	-	-	-	-
%	7.09%		5.32%		3.54%		1.77%	

오류검출 CRC 및 실시간 오류검출 여부								
#	Type 1		Type 2		Type 3		Type 4	
	오류검출	실시간	오류검출	실시간	오류검출	실시간	오류검출	실시간
1	CRC8, 9	X	(CRC0,1),...,(CRC0,9)	O	(CRC0,1),(8,9),...,(8,15)	O	(CRC1,2),(8,9),...,(8,15)	O
2	CRC10, 11	X	(CRC1,2),...,(CRC1,9)	O	(CRC0,2),(8,9),...,(8,15)	O	(CRC1,3),(8,9),...,(8,15)	O
3	CRC11, 12	X	(CRC2,3),...,(CRC2,9)	O	(CRC0,3),(8,9),...,(8,15)	O	(CRC1,4),(8,9),...,(8,15)	O
4	CRC12, 13	X	(CRC3,4),...,(CRC3,9)	O	(CRC0,4),(8,9),...,(8,15)	O	(CRC1,5),(8,9),...,(8,15)	O
5	CRC13, 14	X	(CRC4,5),...,(CRC4,9)	O	(CRC0,5),(8,9),...,(8,15)	O	(CRC1,6),(8,9),...,(8,15)	O
6	CRC14, 15	X	(CRC5,6),...,(CRC5,7)	O	(CRC0,6),(8,9),...,(8,15)	O	(CRC1,7),(8,9),...,(8,15)	O
7	CRC15, 16	X	(CRC6,7),...,(CRC6,9)	O	(CRC0,7),(8,9),...,(8,15)	O	(CRC1,8),(8,9),...,(8,15)	O
8	-	-	(CRC7,8),(CRC7,9)	O	(CRC0,8),(8,9),...,(8,15)	O	(CRC1,9),(8,9),...,(8,15)	O
9	-	-	(CRC8, 9)	O	(CRC0,9),(8,9),...,(8,15)	O	-	-
%	8.86%		11.39%		15.95%		14.18%	

서 비트 오류가 발생할 경우의 수를 나타내었으며, 2~9항은 오류의 위치를 변경하면서 나머지 UI에서 한 비트오류가 발생할 경우의 수를 나타내었다. Type 3~Type11은 서로 다른 DQ 혹은 UI 간에 발생 가능

한 오류를 분류하였다. 즉 Type 3의 1항은 DQ0에 한 비트 오류가 고정적으로 발생하고 나머지 DQ1~7에서 한 비트 오류가 발생할 경우의 예이며, 이때 DQ1에서 UI1~UI9사이에 한 비트 오류가 발생할 경우의 수가 9가지이며, DQ2~7모두를 고려해야하므로 9x7=63가지의 경우가 발생한다. Type3의 2항은 고정 비트 오류의 위치를 DQ1으로 이동시키고 동일하게 적용시 9x7=63가지의 경우가 발생한다. 이와 같은 방식으로 Type 3는 총 9x7x8=504가지의 경우의 수를 가진다. Type 4~Type11도 Type 3와 동일한 방식으로 확장하여 경우의 수를 얻는다. 전체 Type의 총 경우의 수는 280+360+504+448+92+336+280+224+168+112+56 = 3160으로 총 8개의 DQ와 10개의 UI에서 2비트 오류가 발생할 조합의 수 C(8x10, 2)=3,160개와 일치한다. 분류한 이중오류에 대한 실시간 오류 검출율 및 검출범위를 표 4에 나타내었다.

표 4에서 보다시피 전체 이중고장에서 Type1의 경우는 실시간 오류검출이 불가능하여 91.1%의 실시간 오류 검출이 가능하다. 실시간 오류검출이 불가한 경우CRC8~16에서 오류검출이 가능하도록 하여 보안하였다. 삼중고장의 경우의 수는 C(8x10, 3)=82,160 이며 실시간 오류검출이 모두 가능하다.

표 5에 제안한 방식의 실시간 오류 검출 능력을 기존코드방식과 비교하여 나타내었다. 표에서 이중고장 검출율은 제안한 방식에서 CRC8~CRC15를 추가했을 때의 총 검출능력을 나타낸 것이다.

Table 5 error detection capability of proposed CRC scheme
표 5. 제안한 실시간 matrix scheme의 총 오류검출능력

CRC Size (bit)	CRC	Bits Error에 대한 검출 능력				
		1bits	2bits	3bits	4bits	5bits
12	CRC-12	100%	100%	100%	99.96%	100%
8	DARC-8	100%	97.9%	100%	99.87%	99.95%
8	CRC-8	100%	100%	100%	99.81%	100%
16	실시간 matrix	100%	100%	100%	99.90%	100%
7	CRC-7	100%	100%	99.7%	99.83%	99.89%

III 결론

본 논문에서는 기존의 CRC 코드방식을 개선할 수 있는 CRC 비트 구성을 위한 데이타 프레임과 CRC 코드방식을 제안하였고, 실시간 오류 검출이 가능하도록 제안하였다.

기존의 방식은 데이터 비트가 모두 입력된 후에 데이터 비트 오류여부를 판단하여 검출이 가능하지만 제안한 방식은 데이터 비트가 입력되는 즉시 오류검출이 가능하도록 CRC와 데이터 비트를 새로이 제안하였다.(그림 2) 이와같이 실시간 오류 검출이 가능하도록 함으로서 그림 5에서 보다시피 첫 번째 데이터 비트에서 오류가 발생할 경우 오류 검출로 인한 기존 방식 대비 87.5%의 개선효과가 있다. 물론 데이터 마지막 비트에서 오류가 발생할 경우 개선효과는 0%이다. 데이터 비트의 오류가 발생할 확률은 모든 비트에 고르게 분포한다고 가정할 경우 실시간 오류 검출로 인한 평균 개선효과는 ~50%이다. 데이터 비트 오류의 경우의 수에 대한 조합을 정리하여 각 경우에 대한 실시간 검출 가능 여부를 정리하여 표 3~4에 나타내었으며, 비트 오류에 대한 검출율을 표5에 나타내었다.

기존의 DDR4에서 read/write CRC구성을 위해 XOR 6단의 약 700 XOR 게이트를 사용하는데,[2,5] 제안한 방식은 회로구성을 위해 4단을 사용하여 XOR 단의 지연시간을 33% 개선할 수 있으며(표 6), ATM-8 HEC 코드 방식보다 좋은 오류 검출율을 가짐을 표 5에 나타내었다. 또한 제안한 방식은 데이터가 10UI 모두 입력된 이후에 오류 검출하는 방식을 개선하여 매 UI에 데이터가 입력될 때 마다 실시간 오류 검출이 가능하도록 하여 전체 데이터 중 오류가 앞 부분에서 발생할 경우 더욱 개선효과가 증가한다. 따라서 제안한 실시간 오류검출 방식은 고속 반도체 메모리에 적용시 CRC로 인한 SDRAM 내부 동작 마진개선에 효과적이다.

Table 6. error detection rate and overhead of the proposed scheme
표 6. 제안한 방식의 오류검출 속도 및 overhead

구분	CRC		오류검출속도	Area Overhead
	방식	XOR 단	실시간 검출	
기존	ATM-8 HEC	8	X	720 gates
	Moon	6	X	700 gates
제안	실시간 matrix	4	평균50%개선	272 gates

References

[1] D. Graham-Smith, "IDF: DDR3 won't catch up with DDR2 during 2009," in PC Pro, Aug. 2008.
[2] Kibong Koo, et al., "A 1.2V 38nm 2.4Gb/s/pin 2Gb DDR4 SDRAM with Bank Group and x4 Half-Page

Architecture" IEEE International Solid State Circuits Conference, pp. 40 - 41, Feb. 2012.
[3] S. Yoon, B. Kim, Y. Kim, B. Chung, "A Fast GDDR5 Read CRC Calculation Circuit with Read DBI Operation," IEEE Asian Solid-Sate Circuits Conference, pp. 249-252, November, 2008.
[4] Seung-Jun Bae, Kwang-Il Park, "An 80 nm 4 Gb/s/pin 32 bit 512 Mb GDDR4 Graphics DRAM With Low Power and Low Noise Data Bus Inversion," IEEE Journal of Solid-State Circuits, Vol.43, pp. 121-131, January, 2008.
[5] J. Moon. "Fast Parallel CRC & DBI Calculation for High-speed Memories:GDDR5 and DDR4" Circuits and Systems (ISCAS), 2011 IEEE International symposium, pp. 317 - 320, May. 2011.
[6] H. Sheidaeian and B. Zolfaghari, "Parallel Computation of CRC Using Special Generator Polynomials" International Journal of Computer Networks & Communications(IJCNC), Vol.4, No.1, pp39-47 January 2012.
[7] Kyomin Sohn, et al., "A 1.2V 30nm 3.2Gb/s/pin 4Gb DDR4 SDRAM With Dual-Error Detection and PVT-Tolerant Data-Fetch Scheme" IEEE Journal of Solid State Circuits, Vol.48, pp. 168-177, Jan. 2013.
[8] Koopman, P. and Chakravarty T. "Cyclic Redundancy Code(CRC) Polynomial Selection For Embedded Networks" International Conference on Dependable Systems and Networks, pp. 145-154 28 June-1 July 2004.
[9] H. Bui, Y. Wang, and Y.Jiang, "Design and Analysis of Low-Power 10-Transistor Full Adders Using Novel XOR-XNOR Gates," IEEE Transactions on Circuits and Systems II, Vol. 49, No. 1, pp. 25-30, January, 2002.
[10] K. Lin, C. Wu, "A Low-cost Realization of Multiple-input Exclusive-OR gates," ASIC Conference and Exhibit, Proceedings of the Eighth Annual IEEE International, pp.307-310, September. 1995.
[11] JEDEC STANDARD, "DDR4 SDRAM", JEDEC Solid State Technology Association, JESD79-4, September 2012.

BIOGRAPHY

Lee Joong-Ho (Member)

1988 : BS degree in Electronics & Computer Engineering, Ulsan University.
1990 : MS degree in Electronics & Computer Engineering, Ulsan University.
1994 : PhD degree in Electronics & Computer Engineering, Ulsan University.
1994~2012 : SK-Hynix Semiconductor
2012~Present : Professor, Computer Science, Yongin University